

Technical Report

Mini UNIX Shell Implementation

Gaurav Jain (2025MCS2121)

Aman Singh (2025MCS2122)

December 9, 2025

Abstract

This report documents the design and implementation of the Mini UNIX Shell, a custom command-line interpreter developed in C. The system implements a Read-Eval-Print Loop (REPL) capable of tokenizing user input, managing process execution via system calls (fork, execvp), handling I/O redirection, and orchestrating inter-process communication using pipes.

1 System Architecture

The shell is modularized into four primary components, ensuring separation of concerns:

- **Main Controller** (`main.c`): Manages the REPL loop and signal handling.
- **Parser** (`parser.c`): Tokenizes input and builds command structures.
- **Executor** (`executor.c`): Handles process creation and file descriptor manipulation.
- **Builtins** (`builtins.c`): Implements internal shell commands.

2 Data Structures

The core data structure driving the shell is the `command_t` struct, defined in `parser.h`. It utilizes a linked-list approach to support pipelining:

```
typedef struct command
{
    char **argv;           // Arguments array
    char *input_redirect;  // File for '<' redirection
    char *output_redirect; // File for '>' redirection
    int background;        // Background execution flag
    int is_exec;           // 1 = external, 0 = builtin
    struct command *pipe_to; // Pointer to next command in pipeline
} command_t;
```

3 Implementation Details

3.1 Parsing Logic

The parser processes input in two stages:

1. **Tokenization:** The `tokenize` function splits the raw input string into an array of tokens, handling whitespace and special characters like `|`, `<`, `>`, and `&` [?].
2. **Command Construction:** The `parse_tokens` function iterates through the tokens. Upon encountering a pipe (`|`), it finalizes the current command node and allocates a new one linked via the `pipe_to` pointer [?].

3.2 Process Execution

The executor distinguishes between simple commands and pipelines:

3.2.1 Simple Commands

For non-piped commands, the shell calls `fork()`. The child process sets up redirection using `dup2()` if necessary, and then replaces its image using `execvp()`. If the `background` flag is set, the parent does not `wait()` immediately; otherwise, it waits for the child to terminate [?].

3.2.2 Pipeline Management

Pipeline execution is handled by `execute_pipeline`. It iterates through the `command_t` list:

- A pipe is created using `pipe()` for every command (except the last).
- **Child Process:** Binds `STDIN` to the read-end of the previous pipe and `STDOUT` to the write-end of the current pipe using `dup2`.
- **Parent Process:** Closes the pipe ends it no longer needs to prevent file descriptor leaks and hangs [?].

3.3 Signal Handling

To provide a robust user experience, the shell modifies default signal behaviors:

- **SIGINT (Ctrl+C):** Ignored in the parent process (`SIG_IGN`) to prevent the shell from exiting when the user interrupts a subprocess [?].
- **SIGCHLD:** A handler reaps zombie processes using `waitpid(-1, NULL, WNOHANG)`. This is critical for cleaning up background processes that finish asynchronously [?].

4 Conclusion

The Mini UNIX Shell successfully implements core OS concepts including process management, I/O redirection, and IPC. The use of a linked-list structure for commands allows for flexible pipeline construction, while dedicated signal handlers ensure stability during execution.

5 Appendix

Commit History

```
commit cef8b5e29e5b30d5fa270c08364b8ef08804a6
Author: gauravjain2 <43726919@gauravjain2@users.noreply.github.com>
Date: Mon Dec 8 23:35:21 2025 +0530

    implemented executor

commit 5f041f31458335e4c4a4d5044ed764228319567
Merge: f080d09 767688c
Author: Gaurav Jain <43726919@gauravjain2@users.noreply.github.com>
Date: Tue Dec 9 07:45:19 2025 +0530

    Merge pull request #1 from gauravjain2/feature-parser-and-builtins

    Feature: parser and builtins

commit 767688c9da78163f6a14c280c66aebfe58c86 (origin/feature-parser-and-builtins, feature-parser-and-builtins)
Author: Aman Singh <aman.singh2@gmail.com>
Date: Sun Dec 7 20:46:27 2025 +0530

    Add utility functions for directory management and enhance built-in command handling

commit 67f15a584d13fd3cb07326e015448c1478ad56c
Author: Aman Singh <aman.singh2@gmail.com>
Date: Sat Dec 6 18:55:15 2025 +0530

    Implement parser and built-in command functionality

commit f00409041983c579b00118e0d1899c5ef5bedd
Author: Aman Singh <aman.singh2@gmail.com>
Date: Sat Dec 6 15:56:12 2025 +0530

    Initial repo setup

commit a42f0b72a6a824c530504d96a97d1baec9458b0
Author: gauravjain2 <43726919@gauravjain2@users.noreply.github.com>
Date: Sat Dec 6 11:02:17 2025 +0530

    added readme and .gitignore

(END)
```

Figure 1: Commit History 1

```
commit 94f0960033f4b05037824e52ba11f0ef267c3b3
Author: gauravjain2 <43726919@gauravjain2@users.noreply.github.com>
Date: Tue Dec 9 09:52:34 2025 +0530

    updated README

commit e02ba3c73e1d65aab43cebbca8b3eb8ae7475c6f
Author: gauravjain2 <43726919@gauravjain2@users.noreply.github.com>
Date: Tue Dec 9 09:18:03 2025 +0530

    Added user documentation

commit f3a582e528806df5c138427106b17d5583de559f
Author: gauravjain2 <43726919@gauravjain2@users.noreply.github.com>
Date: Tue Dec 9 09:12:15 2025 +0530

    added multistage pipeline support

commit 3f527b2cec3a7eeFeb921204768fcd72cb70a26
Merge: 5f041f3 6c05f00
Author: Gaurav Jain <43726919@gauravjain2@users.noreply.github.com>
Date: Tue Dec 9 08:23:25 2025 +0530

    Merge pull request #2 from gauravjain2/feature/executor

    Feature: Implemented Executor & Integrated with Parser

commit 6c05f002d0b37b0a272681b55154cf1b11a49 (origin/feature/executor, feature/executor)
Author: gauravjain2 <43726919@gauravjain2@users.noreply.github.com>
Date: Tue Dec 9 08:16:54 2025 +0530

    integrated parser and utility with main

commit 5805addc2578e7c4b46709745d080679ba6c220
Author: gauravjain2 <43726919@gauravjain2@users.noreply.github.com>
Date: Tue Dec 9 08:16:07 2025 +0530

    added makefile
```

Figure 2: Commit History 2